Network-Based Authentication: The Key to Security

by Carl Malamud

In most computer networks, determining a person's identity is often done through passwords. But unencrypted passwords are easy prey for people trying to break into your system; for example, protocol analyzers can view the contents of packets as a diagnostic capability. However, alternatives to passwords exist, offering sophisticated methods for computers to protect their conversations from eavesdropping.

Authentication is the process of ensuring the identity of another object (user, program or computer). For example, when a client program uses a distributed database server, it is important to verify that the client is intended to have access to the requested data. Since access to data is most commonly based on the identity of a user, authentication protocols provide a secure way of determining that identity, preventing masquerades and other unauthorized attempts to access data.

Since passwords are a primitive way of providing authentication, in this article we look at public key cryptography, a sophisticated way for two computers to authenticate each other and then protect their conversation from eavesdropping.

So What's Wrong With Passwords?

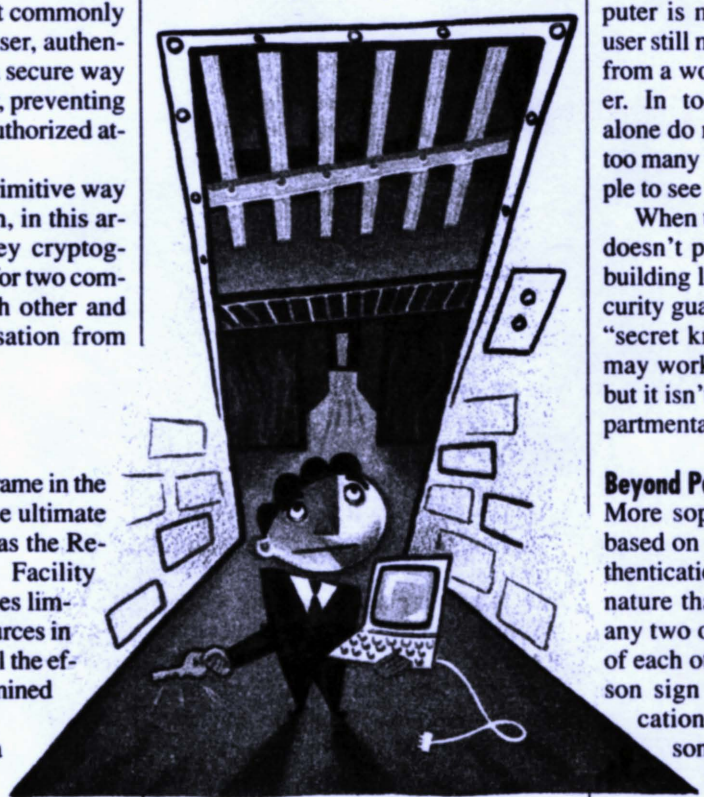
We often think of the mainframe in the glass house as providing the ultimate in security. Products such as the Remote Access Control Facility (RACF) for IBM mainframes limit access to computing resources in ways that are intended to foil the efforts of even the most determined hacker (or "cracker").

On the other hand, a workstation appears to be a security manager's nightmare. Workstations are often managed by inexperienced people working with easy-to-use operating systems such as DOS, Macintosh, or even Unix (that is, Unix being "easy-to-use" only in comparison to MVS).

As it turns out, the reverse is actually true: The workstation is often safer than the mainframe.

Here's why: The mainframe has the potential to be an extremely safe environment. Even after getting on the system, all sorts of access controls and privilege levels limit what a particular user can do. The weak point, however, is system access. Theoretically, cracking most password algorithms requires some pretty sophisticated processing. In reality, however, there are two ways that passwords are made vulnerable: social engineering and network protocol analyzers.

Most crackers use a technique known as social engineering. Your password



can often be determined by trying (in order) the name of your spouse, your dog and your car. Often, a hacker can simply walk by your office and look for a note on your terminal; chances are, it's your password.

System administrators can use a variety of techniques to govern how passwords are used. Minimum-length requirements avoid short, easily guessed passwords. Password aging requires a password to be changed periodically (and ensures that a prior password cannot be reused). Soundex algorithms make sure that the password is not a common-sense word. Some sites even require a mix of numbers, upper-case and lower-case letters.

Even with well-formed passwords, a problem remains: The password has to be transmitted from the user to the system. Network analyzers, such as Novell's LANalyzer, provide an easy way to trap such information. Even if a computer is not on a public network, the user still needs to transmit the password from a workstation to another computer. In today's networks, passwords alone do not provide enough security; too many tools exist to allow other people to see passwords.

When the CIA has a lot of secrets, it doesn't put all of the secrets in a big building labeled SPIES, then post a security guard at the door waiting for the "secret knock. "Shave and a haircut" may work fine for general admission, but it isn't the best way to handle compartmentalized security problems.

Beyond Passwords

More sophisticated access control is based on authentication schemes. Authentication is an encrypted, digital signature that allows two computers (or any two objects) to verify the identity of each other. Much like having a person sign a piece of paper, authentication allows us to verify the person's identity (and thus hold that person to any promises he or she has made). This digital signature is thus like a very long, very complex password.

One example of authentication is having two electronic mail users communicate. Another is protecting a naming service such as Novell's NetWare,

ROBERT DEMICHIEL

where it is important to ensure that users can't see each others' passwords. Authentication protocols can even be used to provide secure voice communications and font encryption.

Authentication schemes typically use encryption. This conceals data by conducting computations against a key—an alphanumeric string. Encryption algorithms range from fast, simple methods to more complex operations involving long keys and difficult computations.

There are currently two forms of authentication in use: symmetric key and public key. Symmetric key, typified by the MIT-developed Kerberos system, requires two users to share a key. That key is then used with some form of encryption algorithm, such as the U.S. Government-sponsored DES (the Data Encryption Standard), which is widely implemented both in hardware and software.

Once two users share a symmetric key, they can encrypt and decrypt traffic on the network. In order to start communicating, the initiating user forms a "ticket," comprising the user's credentials. A typical ticket has the name of the user, the date that the ticket was formed and the date that the ticket expires. All this information is encrypted using the symmetric key and sent over the network. The program on the other side applies the symmetric key and decrypts the credentials.

Putting timestamps into a credential prevents a problem known as session replay. With replay, an unauthorized user puts a protocol analyzer on the network and captures the credentials. Later, the credentials are sent (or replayed) to an unsuspecting target, allowing the hacker to impersonate a legitimate user. Timestamps make the credential worthless after a short period of time.

The advantage of symmetric key schemes like DES is that they are relatively fast. But key distribution centers are the weak link in systems like Kerberos. The key distribution center is the computer that passes out the key to the two computers that wish to communicate with each other. This is the classic bootstrap problem: If the key distribution center is also on the network, how do we protect the information it disperses?

International standards, such as X.509, use an alternative method known as public key encryption. Public key encryption also forms the emerging standard for privacy and authentication in research networks based on the TCP/IP protocols. The initial application for OSI networks (which use X.509) and TCP/IP is in electronic mail.

Rather than sharing a single secret key between two users, public key encryption uses two keys: a public key and a private key. Each user receives a private key. The public key is stored someplace on the network, accessible to all other users.

Sophisticated authentication protocols are key to network-based security. Passwords are easily stolen or broken, so network security typically includes cryptography techniques.

To encrypt something, the public key is used; messages are then decrypted using private keys. Anybody can encrypt a message using the public key, but the only way to read it is with a private key. Public and private keys are the inverse of each other: anything encrypted with one can only be decrypted with the other. A message encrypted with a public key cannot be decrypted with the same key; a similar limitation applies to private keys.

The value of public key encryption is in the difficulty of reversing the algorithm. Since public key algorithms are based on factoring, the basic decryption problem is how to take a large number and break it up into its factors. The computational difficulty of breaking a key is typically expressed in Mip-years: how many years of CPU time would a 1-Mip machine take to break a key?

A recent example of key-breaking was conducted by a team of researchers led by A. Lenstra. The team used an ingenious algorithm to break one of the simplest possible 155-digit numbers: two 1's with 153 zeros in the middle. Even this "simple" task took over 250 Mip-years. More complex 155-digit numbers can easily require millions or tens of millions of Mip-years to be broken.

Most public-key schemes use an algorithm developed by RSA Data Security. The keys in this scheme, which can range up to 512 digits long, can be significantly harder to break than keys generated with other schemes. For example, one of the public keys used by people communicating with RSA is 301 digits long, requiring over one trillion Mip-years to break, given the current state of research in number theory.

Public keys are stored on nameservers throughout the network. Since public keys are not sensitive information, they can be freely distributed. The private key is generated on your workstation and stays there; it never enters the network. While public key cryptography is safe, it is also somewhat slower than many other schemes: public key encryption takes 100 to 1,000 times as long as symmetric encryption. If every single packet needs to be authenticated, public key cryptography would exact an undue performance burden.

The Keys of the Internet

As a result, most groups tend to use a combination of public and symmetric methods. In such a scheme, public key cryptography is used to encrypt a DES key. The recipient uses a secret key to decrypt the credentials, extracting the DES key. This key is then used to decrypt the actual message. Because the DES key is randomly generated for each transaction, and is then encrypted, the system is quite secure.

The Internet, the worldwide research network, uses an example of this combination of public and symmetric key methods. The Internet's new Privacy Enhanced Mail (PEM) standards sit on top of the regular mail protocols and do not require any changes to the underlying installed base. Public key credentials, available to anybody, are still employed. In addition, each PEM user generates and secures an RSA private key.

To send electronic mail, a DES key is used to encrypt the actual message. The public key is then used to encrypt the DES key. Note that in this scheme both authentication and encryption are used at the same time. Authentication uses encryption to protect the credentials, ensuring that the credentials remain safe. Encryption of the actual message serves another purpose: protecting the message itself from unauthorized viewing.

In addition to message encryption,

the message includes another piece of information: the Message Integrity Check (MIC). The MIC is a unique 128-bit quantity based on the contents of the message. Like a cyclic redundancy check (CRC), the MIC is very different if even a single bit of the message has been altered.

Though the MIC is a way of ensuring that somebody has not played with the message while in transmission, it has another important role: non-repudiation. Because the MIC is encrypted using the RSA private key, we know that only the real sender of the message could have generated a particular MIC for a particular message. Assuming a person's RSA private key was not stolen, we can be virtually certain that person is really the one who sent the message. This fact can't be denied later. In function, this is similar to a credit card: If a card isn't reported as stolen, the owner is responsible for charges on the card.

Built-In Authentication

Authentication is clearly better than password-based control alone, providing a more sophisticated way of handling things than a password. Think of the password as a key being sent in unencrypted form. A password is typically only eight to 12 characters long, is managed by a single person, and is transmitted over the network wire. In contrast, an RSA public key is available to all people, while the private key, equivalent to the password but mathematically more sophisticated, is virtually "unguessable" and never enters the network.

In order to use authentication, however, you really need a computer. You're not going to remember a 512-bit quantity, let alone generate the encryption, message integrity checks, and other aspects of a privacy-enhanced computing environment without some computational help.

Since people are unable to remember their private keys, most systems store a person's private key encrypted by a password—not the best system because the first computer in the chain is still vulnerable to a password attack. This process does, however, prevent passwords from ever showing up as clear text on the network, which remains one of the biggest security threats. Of course, smart cards, which place electronically encoded information on a credit card-sized key, will eventually change this, allowing people to carry their private keys around with them. But even a smart card tends to require a personal identification number (equivalent to a password) to ensure against loss.

Distributed Security

Security in the distributed world is based on the client-server model of computing. Servers provide access to data for their clients, plus a host of other services. However, the data that a client leaves on a host may be encrypted so that the server and its other users have no idea what the data means.

Clients, on the other hand, can allow public access to a few portions of their local file systems and seal off others. Private keys are stored in private areas.

On a centralized, timesharing system, once you've broken the initial front door, you've gone a long way toward compromising the security of many other users. On a workstation, you've only compromised one user. If you want to access another workstation's data, you have to start from scratch and break into the new system.

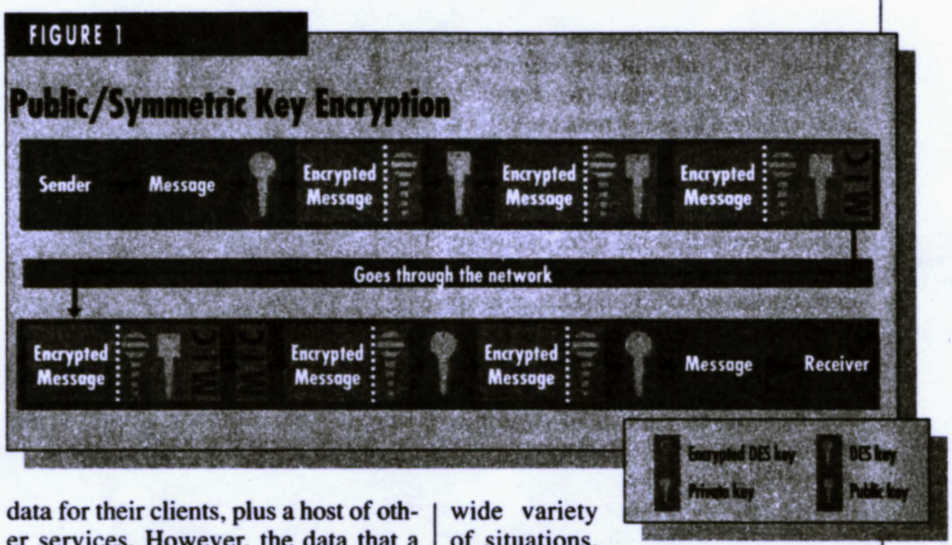
But there's one potential problem in such a chain. While authentication works between computers, we still need to authenticate the user on the first computer. The current method of doing is this is a password.

If passwords are going to work on a workstation, the workstation must be physically secured. There needs to be a way to prevent unauthorized users from wandering around and rebooting workstations in superuser mode, such as locking the computer in a room. Physical security, smart cards, proper password administration and other management techniques are the way to handle this problem, distributing the risk of a break-in.

Authentication makes each computer a separate security realm. First, RSA public key cryptography allows two computers to authenticate each other; then DES encryption preserves the confidentiality of communications in a particular session.

Authentication based on public key cryptography is starting to get used in a wide variety of situations. Lotus, for example, uses the RSA protection mechanisms in Notes, its groupware software for conferencing. The same technology is also used in Novell's NetWare as part of its new naming services.

FIGURE 1



While Notes and NetWare are two fairly obvious uses for the RSA products, there are some less intuitive applications. Tektronix, for example, uses public key cryptography to protect fonts on their printers. Motorola uses the same technology to provide secure voice communications, encrypting entire conversations. Even smart cards can use public key cryptography to authenticate a card to the device that is reading it (and vice versa).

Companies like RSA are beginning to address the security infrastructure on networks, making large public networks like the Internet more secure. The same security infrastructure is being used in distributed workgroups, with software such as Notes and NetWare. Increased and improved security, rather than closing off networks, is making them more trusted and thus more usable. ■

Carl Malamud writes professional reference books. He can be reached at carl@malamud.com (Internet) or cmalamud (MCI Mail).